代码说明

一、 需求背景

远程共享和工作协同目前发展比较成熟。其中协同设计、协同著作、远程教学等是这些应用的典型代表。对于设计的人来说，接触到的产品就是电子白板，手持 ipad 绘画插图，传输到打印机或者 PC；喜欢上网络课程的，相信大家也接触过沪江网校，里面就是共享的实现；对于喜欢看直播的人，经常看见主播闲暇时玩互动游戏你画我猜，你画我猜就是一个白板共享。这些都是比较好的案例。

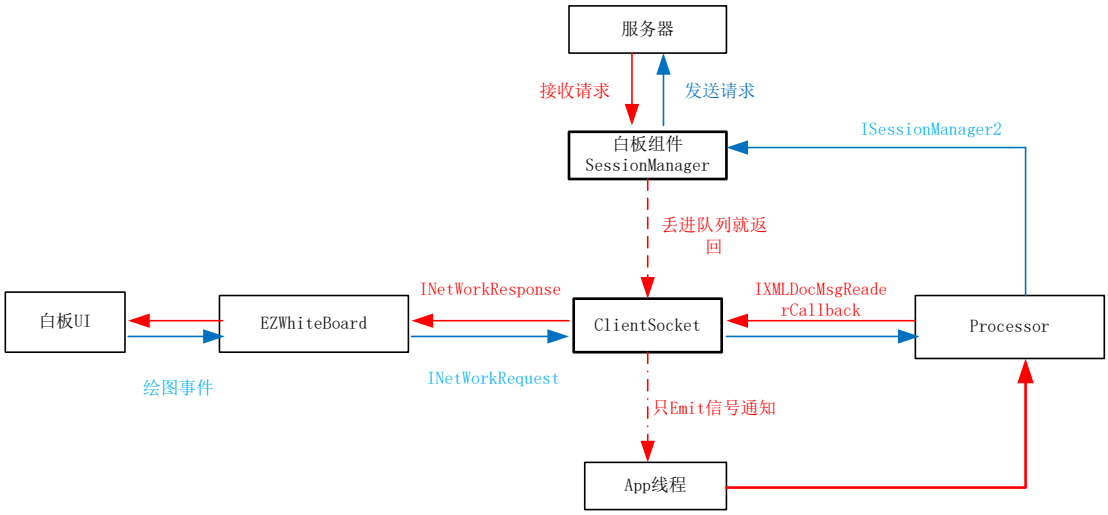
因此，作为一款视屏会议产品，共享白板的是必不可少的。ezTalks cloud 需要实现白板共享，白板操作，和参会人白板权限等问题。

二、 设计思路

1）共享空间的管理与维护，提供有效的控制机制，防止对现有工作的恶意破坏。简单讲就是权限问题，不同权限对白板有不同的操作，通常情况下都是，一个人对白板写操作，其他人员只有读操作，可以将其描述为“one write, other read”。这只是最基础的权限管理，后续提升互动性能，权限将会更复杂。参考代码权限枚举，和权限设置函数 setaccessmode。

2）白板显示的通信问题，设计电子白板的目的在于能更好地进行异地或本地的协同工作，如果参与者之间存在信息“鸿沟”或者误差，那就失去了意义。共享白板系统采用的通信协议集是在标准的 TCP/IP 协议族基础上的应用层协议。采用 C/S 模型。使用白板组件，不必考虑网络库问题，制定规范和高效的协议是主要工作。可以参考白板说明文档协议相关部分。

3）整体框架，大致如下。



EZWhiteBoard ---------- 白板管理器

ClientSocket ---------- 事件管理器

Processor ---------- 事件执行器

核心部分为 EZWhiteBoard 白板管理器和 ClientSocket 事件管理器。

1、 主讲共享时绘图事件从白板 UI 开始，通知白板管理器，白板管理器调用事件管理器，事件管理器组装协议后，用 Processor 发送给服务器。蓝色流程

2、 其他用户接受共享时，由组件把来自服务器的事件丢进管理器队列，事件管理器有一个线程不断取事件，并且通知 App 线程处理（emit，事件管理器对 App 开放事件处理接口），最终事件由 App 异步消耗。红色流程

三、 实现说明

1、 模块之间接口耦合。

2、 ClientSocket 接受服务器请求时，异步处理。

3、 白板管理器观察者。

四、 代码文件

1、只拿 **clientsocket.cpp**，其他是相关的代码

```
class ClientSocket : public WThread                    //线程功能
                    ,public MsgHandler                 //消息处理接口，App 使用
                    ,public IXMLDocMsgReaderCallback   //回调接口，Processor 回调 ClientSocket
                    ,public INetWorkRequest            //网络请求接口，供 EzWhiteBoard 使用
```

WThread：

1、 在会议室初始化时，初始化白板组件，同时开启线程。线程主体执行函数如下

```
244  ************************************************************************************************/
245  FS_UINT32 ClientSocket::ThreadProcEx()
246  {
247      while(m_bRun)
248      {
249          CMsgMpNode *pMsg = NULL;
250          pMsg = m_NetMsgQueue.PopMsg(1000);
251          if(pMsg && m_fcMainThreadCall)
252          {
253              m_fcMainThreadCall(this,pMsg);//取到数据丢到主线程处理
254          }
255      }
256      return 0;
257  }
```

这里从消息队列中不断取出，通过 m_fcMainThreadCall(this,pMsg)通知主线程。

2、m_fcMainThreadCall(this,pMsg)函数指针实际注册的是如下函数，作用是向 UI 线程发送一个信号，同时给出自己的消息处理接口指针 MsgHandler，由 UI 线程处理来自服务器的事件。

```
80  void EZSTDCALL birdgeMsgToUiThread(MsgHandler *target,void *msg)
81  {
82      emit ezApp->signal_toUIThread(target, msg);
83  }
84
```

MsgHandler：

1、 只有一个接口函数 handleMsg，开放给 UI 调用的。

```cpp
bool ClientSocket::handleMsg(void* pMsg)
{
    if(pMsg == NULL)
        return false;

    CMsgMpNode *p = static_cast<CMsgMpNode*>(pMsg);

    if(p && p->GetMessage() == WM_SESSIONNOTIFY)
    {
        SESSION_EVENT *pEvent = NULL;
        while((pEvent = m_pSessionManager->GetEvent(p->GetLParam())) != NULL)
        {
            this->ProcessSessionEvent(pEvent);//具体消息处理
            m_pSessionManager->FreeEvent(p->GetLParam(),pEvent);
        }
        m_NetMsgAllocator.Free(p);
    }

}
```

IXMLDocMsgReaderCallback：

1、 供 Processor 回调的接口。UI 线程处理消息时，进入 Process，然后回调 ClientSocket 相关接口函数解析数据， ClientSocket 再通过 INetWorkResponse 接口调用 EZWhiteBoard，真正执行响应，从而更新白板。

INetWorkRequest：

网络请求接口，封装了白板操作的抽象请求。当白板有绘图事件发生时，通过该接口封装协议打包消息，最后经 Processor 发送给服务器。

五、总结

核心就是回调。

普通回调，设置事件 Sink

虚函数+回调+Qt 信号，委托事件

接口封装供回调